PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/445

A1

(11) International Publication Number: WO 98/54642

(43) International Publication Date: 3 December 1998 (03.12.98)

(21) International Application Number: PCT/IB98/00334

(22) International Filing Date: 12 March 1998 (12.03.98)

(30) Priority Data:
08/866,651 30 May 1997 (30.05.97) US

(71) Applicant: KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL–5621 BA Eindhoven (NL).

(71) Applicant *(for SE only)*: PHILIPS AB [SE/SE]; Kottbygatan 7, Kista, S–164 85 Stockholm (SE).

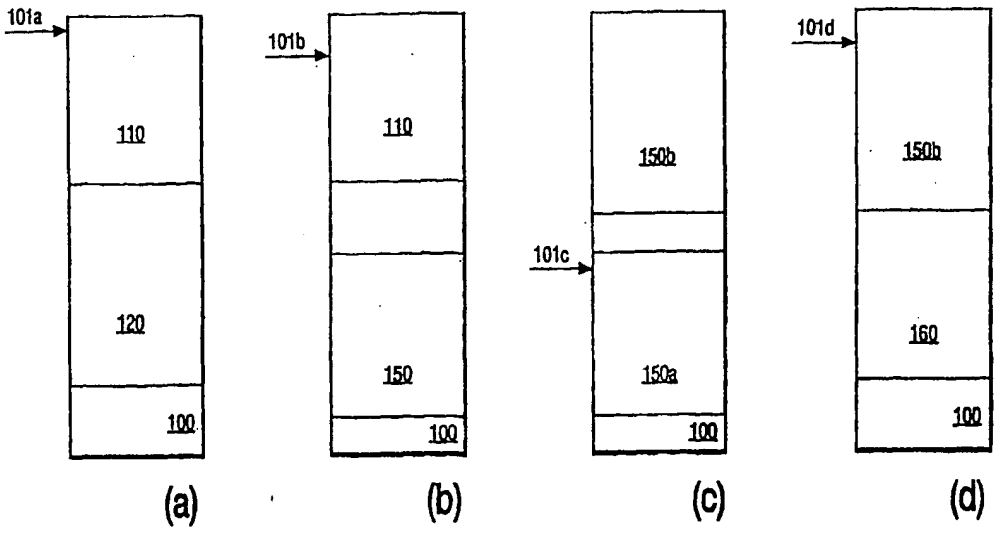
(72) Inventors: RATH, Kamlesh; Prof. Holstlaan 6, NL–5656 AA Eindhoven (NL). WENDORF, James, W.; Prof. Holstlaan 6, NL–5656 AA Eindhoven (NL).

(74) Agent: GROENENDAAL, Antonius, W., M.; Internationaal Octrooibureau B.V., P.O. Box 220, NL–5600 AE Eindhoven (NL).

(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: FAILSAFE METHOD FOR UPGRADING SET–TOP SYSTEM SOFTWARE FROM A NETWORK SERVER



(57) Abstract

A method and device which allow for the failsafe downloading of system software from a network server, without requiring additional memory. The system software is structured to consist of a primary partition and a secondary partition. The primary partition contains the software required to download the secondary partition, as well as the software to download a new primary partition. At all times, a verified copy of either an old or a new primary partition is present in memory, thereby allowing for a reexecution of the downloading process, in the event that the download process is interrupted or the received partition is corrupted.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Fail safe method for upgrading set-top system software from a network server

This invention relates to the downloading of system software from a network server. In particular, cable television providers supply "set-top" devices to manage and control the user access and capabilities. These devices are periodically upgraded by downloading the appropriate software from a network server at the local cable television substation. This invention presents a robust, fail-safe, method for upgrading such set-top devices from a network server.

The downloading of software from a master station to a remote device is common. Protocols establish the specific details for such transfers, and both the master station and remote device must conform to the established protocol to effect the transfer. Traditionally, either the remote device requests the download, or the master station directs or mandates the download. In either event, the remote device must be placed in an appropriate mode to accept the download, after which the transfer of data from the master station to the remote device commences. Typically, the downloaded software is placed in non-volatile memory, such as flash memory, so that it can be locally executed, after power outages or system resets, without reliance on communications with the master station.

Environmental factors can affect the efficiency and reliability of the data transfer to a remote device. At times, the communications link may become disconnected, or so degraded so as to be effectively disconnected, after the transmission has started, but before all the data is received at the remote station. If the data being transferred is software code required for the effective operation of the remote device, the loss of part of the data is often more catastrophic than not having received the data at all. During the download, the new data will be stored in memory, replacing the data that had been stored in that same memory location. The new data will, in all likelihood, be incompatible with the segments of the older data which has not yet been replaced by additional new data. Similarly, the older data will be incompatible with the new segments which have been loaded. Because of this incompatibility, a partial download of new data will often result in a completely inoperative device, unless measures are taken to maintain a full working copy of the old data, for subsequent use, should a partial download occur. This full copy of the old data is not eliminated until it is ascertained that a full copy of the new data has been received. In

practice, the old data is not eliminated, per se, but rather, the set-top box is directed to henceforth execute the data contained in the area where the new data has been stored.

The need to maintain a full copy of the old data while the new data is being loaded requires that twice the amount of memory be provided in the remote device. After a successful download, each half of the memory contains a full copy of the data, the old data in one half and the new data in the other. When a subsequent download is called for, the "subsequent" data is placed in the memory half containing the "old" data, maintaining a full copy of the "new" data in the other memory half until it is ascertained that a full copy of the subsequent data has been received.

Alternatives have been proposed to minimize the amount of extra memory required to effect a reliable download. In European Patent, EP 0524719 A2, for example, the remote device contains a small core of software in permanent memory; if a download is unsuccessful, this small core software is executed to restart the download process. In this way, the only extra memory required is the amount of permanent memory required to contain the small core software. This approach reduces the amount of additional memory required to that required to initiate and control the reboot process. To have optimal effectiveness, this approach requires that the small core software be kept to a minimum, sometimes at the cost of other capabilities. For example, in a typical device containing such a small core software, the device is effectively inoperative until a successful download can be accomplished, because the small core software only contains enough capability to accomplish the download.

The object of this invention is to provide a means of downloading data to a remote device without requiring any additional memory, and yet allowing for the preservation of the code required to initiate and control the reboot process, in the event of a disruption in the download process.

This invention is premised on the observation that downloaded code, such as system software, may be partitioned into two subsets of code: that code which is required for basic, fundamental operations, such as downloading, and that code which is required for non-fundamental operations, such as displaying help menus and the like. Further, by downloading the code in phases, the fundamental operations code can be preserved in full, provided that the fundamental operations code is designed to consume less than half the memory area.

During the download process, the memory associated with the non-fundamental operations is used to store the new fundamental operations code. When the new

fundamental operations code is ascertained to have been downloaded correctly, the new non-fundamental operations code is downloaded into the memory associated with the old fundamental operations code. Should a disruption occur during the first phase, the download of the new fundamental operations code, the old fundamental operations code will be in memory; and, should a disruption occur during the second phase, the download of the new non-fundamental operations code, the new fundamental operations code will be in memory. Thus, despite when a disruption occurs, a copy of the fundamental operations code, including the code required to download either or both portions of the new code, is available for subsequent execution.

Note that no additional memory is required to provide this fail-safe capability. And, compared to the alternatives which maintain a duplicate of a small core of code, this method allows the use of a full half of the available memory for fundamental operations. This would allow the device to perform more functions than merely downloading code. In a cable television set-top device, for example, the processing of user commands, such as channel selection, could be included in the fundamental operations code, thereby allowing the set-top box to be utilized despite a downloading problem.

Figure 1 shows a method for system software upgrades in accordance with this invention.

Figure 2 shows an alternative method for system software upgrades in accordance with this invention.

Figure 1 shows a method for system software upgrades in accordance with this invention. Figures 1(a), 1(b), 1(c), and 1(d) display the state of the memory 100 at four sequential periods of time, respectively.

Initially, at 1(a), the memory 100 contains the old code, which is partitioned into a primary partition 110 and a secondary partition 120. The primary partition 110 contains the fundamental operations code, including the code required to download a new primary partition. Secondary partition 120 contains other, non fundamental operations code, which can be any code which is considered non-fundamental to the operation of the system, and specifically, non-fundamental to the download of new operations code.

When a download of new system software commences, the appropriate code in the old primary partition 110 is executed to load a new primary partition into memory, as shown in figure 1(b). The program pointer 101a is shown to be pointing to the old primary partition 110, signifying that the system is executing code in that area of memory during this time period (a). The new primary partition 150 is loaded into memory

100 in the lower half of memory, including part of the area in which the old secondary partition 120 had occupied, by executing code in primary memory 110, as shown by program pointer 101b. Consistent with this process, a memory location within the old primary partition is updated to indicate that, upon commencement of this download, the old secondary partition is no longer valid, and is considered no longer present in memory 100. At the end of this stage (b), the memory 100 will contain the old primary partition 110, and the new primary partition 150. If a problem occurs during the download of the new primary partition, the old primary partition remains intact in memory 100, to provide for fundamental operations, including repeated attempts to download the new primary partition until the memory 100 contains a verified copy of the new primary partition 150.

Not shown in figure 1 is the means for setting the program pointer 101 (101a, 101b, 101c, and 101d). As is common in the art, the system will contain other non volatile memory which is used to control the fundamental aspects of system management, such as where to initiate program execution after a power outage, or system reset. Consistent with this invention, the appropriate starting location for each of the phases of the process, shown as figures 1(a), 1(b), 1(c), and 1(d), would be contained in this system management memory. Other parameters required to effect a proper restart of the system may also be stored in this memory. For example, this memory would typically contain the location in memory 100 where each partition is to be loaded, the extent of each partition, and the starting address for executing each partition. For ease of understanding, all the parameters required to effect a restart after a system reset will be termed herein as a reset vector. For example, during normal operation (figure 1(a)), before commencing a memory download, the reset vector would contain the address within the primary partition of the start of the routines which provide for normal operation, and any other parameters required to facilitate the start of such normal operations. Upon commencement of a download (figure 1(b)), the reset vector would contain the address within the primary partition of the start of the routines and any other parameters, such as the location and extent of the memory partitions, which are required for the download operation. Should the download process go awry, a system reset will effect a restart of the download operation from the original location using these preset parameters. The reset vector, in accordance with this invention, is not changed until the system verifies that the current phase of the process has been completed successfully, and the next phase is to begin. This can be effected, for example, by loading the reset vector with the next phase starting parameters after the current phase is verified as having been completed, and then forcing a system reset. Thenceforth, each system reset will force the

start of that next phase, until that phase is verified and the subsequent phases starting parameters are placed in the reset vector. As is evident to one skilled in the art, this updating of the reset vector must be performed as a single, all or nothing, operation. That is, to assure a fail safe download, the reset vector must either be updated with all the new starting parameters, or left as is, containing the old starting parameters. Such all or nothing operations, which either complete the entire operation or have no effect if interrupted, are commonly termed atomic operations. Multi-step, non-atomic, updates which could cause the reset vector to contain neither the full set of old nor the full set of new starting parameters should not be employed, for fear that a mishap during this non-atomic update will result in neither the old nor the new code being executed properly. The minimum information to be contained in this reset vector is an indication of where to initiate the execution of code corresponding to the phases depicted as 1(a), 1(b), 1(c), and 1(d) in figure 1; and, as will be subsequent discussed, phases 2(a), 2(b), and 2(c) depicted in figure 2. The implementation of an operation to update such a minimal reset vector in an atomic fashion is common in the art.

As would be apparent to one versed in the art, alternative methods of setting the program pointer 101 may be employed. For example, the memory 110 may be structured as banks, or blocks, of memory, the reset vector parameters corresponding to the execution of each partition could have a fixed location within one of the banks of the partition, and a reset operation could be structured so as to always start at that fixed location within a selected bank. Changing the reset parameters, including the program pointer 101, would be effected by merely changing the bank which is selected to be active upon reset.

After the new primary partition 150 is verified as having been loaded into memory 100, the reset vector is atomically updated, and the code in the new primary partition 150 is executed to continue with the download process, as shown in figure 1(c) by program pointer 101c. A copy of the code 150 is loaded into the area of memory previously containing the old primary partition 110. For clarity, the downloaded code 150 is shown as 150a in figure 1(c), and the copy of downloaded code 150 is shown as 150b. After the copy sequence is completed and verified, the code in 150b is executed to continue the downloading process, as shown by program pointer 101d in figure 1(d). The new secondary partition 160 is downloaded into the memory 100, in the area previously occupied by the old secondary partition 120, and the downloaded new primary partition 150a. As shown in figure 1(d), at the conclusion of this phase, a new primary and secondary partition has been downloaded into memory without consuming any additional memory for the process.

Note that, as shown in figure 1, the size of each of the old and new partitions can be such that the boundaries between primary and secondary old and new partitions need not be the same. That is, the new and old partitions can be of differing sizes, provided only that the total memory consumed by the partitions shown at the states of time represented by figures 1(a), 1(b), 1(c), and 1(d) are each less than the total memory space available in memory 100. As shown in figure 1(c), two copies of the new primary partition are required to be in memory at the same time. As such, in accordance with this invention as thus far disclosed, the new primary partition must be less than or equal to half the total available memory.

As shown by this invention, at any of the states represented by figures 1(a), 1(b), 1(c), 1(d), at least one copy of a verified primary partition is available throughout the downloading process. Thus, should a problem develop during any of the downloading or copying processes between states, a verified primary partition is always available to repeat the corrupted or aborted process.

An alternative embodiment of the invention is shown in figure 2. Items in figure 2 having similar functions to those in figure 1 are referenced by the same numerals. In this embodiment, the new primary partition 150 is loaded into the opposite extreme of memory as the old primary partition 110. That is, in conventional terminology, if the old primary partition is loaded at the lower part of memory 100, the new primary partition is loaded at the upper part, and vice versa. The memory constraint in this implementation is that the sum of the old and new primary partition sizes must not exceed the total memory available. Typically, each of the old and new primary partitions will be limited to half the total memory available, in order to conform to this constraint. As would be evident to one skilled in the art, however, configurations could be employed with greater than half the available memory being allocated to one of the old or new primary partitions, provided the corresponding new or old primary partitions are equivalently less than half the available memory.

Upon verification that the new primary partition 150 has been loaded into memory 100, as shown in figure 2(b), the code in partition 150 is executed to download the new secondary memory partition 160 into the remaining available memory in memory 100, as shown in figure 2(c). The sum of the primary and secondary partitions will have been designed to be less than or equal to the total memory available in memory 100.

As in figure 1, at any of the states shown in figures 2(a), 2(b), and 2(c), a verified version of a primary partition, either old (110) or new (150), is available in memory

100 at all times, so that an interrupted or aborted download can be reinitiated by executing the appropriate code in this verified primary partition.

The foregoing merely illustrates the principles of the invention. It will thus be appreciated that those skilled in the art will be able to devise various arrangements which, although not explicitly described or shown herein, embody the principles of the invention and are thus within its spirit and scope.

CLAIMS

1. A method for downloading code to a device having a memory, said memory having a first area and a second area, said code having a first segment and a second segment, said device being configured to download the first segment of code by executing the code located in the second area of the memory, comprising the steps of:

downloading the first segment of code to the first area of the memory,

verifying that the first segment of code has been downloaded successfully, and,

if the first segment of code has been downloaded successfully:

configuring the device to download the second segment of code by executing the code located in the first area of the memory, and

downloading the second segment of code to the second area of the memory.

2. A method as claimed in claim 1, wherein said first and second areas of the memory are the first and second halves of the memory, respectively.

3. A method as claimed in claim 1, wherein the step of configuring the device to download to the first or second area of memory comprises the step of setting a starting address to an address within the second or first area of the memory, respectively.

4. A method as claimed in claim 1, further comprising the step of:

repeating the step of downloading the first segment of code if the first segment of code is not verified to have been downloaded successfully.

5. A method as claimed in claim 1 for downloading code to a device having a memory, said memory having a first area and a second area, said code having a first segment and a second segment, said device being configured to download the first segment of code by executing the code located in the second area of the memory, comprising the steps of:

downloading the first segment of code to the first area of the memory,

verifying that the first segment of code has been downloaded successfully,

and,

if the first segment of code has been downloaded successfully:

configuring the device to copy the first segment of code to the second area of the memory by executing the code in the first area of the memory,

copying the first segment of code to the second area of the memory, and,

verifying that the first segment of code has been copied successfully, and

if the first segment of code has been copied successfully:

configuring the device to download the second segment of code by executing the code located in the second area of the memory, and

downloading the second segment of code to the first area of the memory.

6. A method as claimed in claim 5, wherein said first and second areas of the memory are the first and second halves of the memory, respectively.

7. A method as claimed in claim 5, wherein the step of configuring the device to copy to the second area of memory comprises the step of setting a starting address to that of the first area of the memory.

8. A method as claimed in claim 5, wherein the step of configuring the device to download to the first area of memory comprises the step of setting a starting address to that of the second area of the memory.

9. A method as claimed in claim 5, further comprising the steps of:

repeating the step of downloading the first segment of code if the first segment of code is not verified to have been downloaded successfully, and,

repeating the step of copying the first segment of code if the first segment of code is not verified to have been copied successfully.

10. A programmable device, said device comprising:

a memory,

said memory comprising a first area and a second area,

operational code,

said operational code having a first segment and a second segment,

said first segment of code being located in said first area of the memory,

said second segment of code being located in said second area of the memory,

said first segment of code comprising a means for downloading a first new segment of a new operational code into the second area of the memory,

means for verifying the download of the first new segment into the second

area of the memory,

said first new segment of code comprising a means for downloading the second new segment of code into the first area of the memory, and

means for executing the first new segment of code for downloading the second new segment of code in dependence upon the verification of the download of the first new segment.

11. A device as claimed in claim 10, further comprising:

a second memory,

said second memory comprising a reset vector,

said reset vector containing one or more parameters for activating the means for downloading the first new segment, or for activating the means for executing the first new segment of code, in dependence upon the means for verifying the download of the first new segment of code.

12. A device as claimed in claim 11, wherein the parameters contained in the reset vector comprise an old set of parameters, said device further comprising

means for loading a new set of parameters into the reset vector,

said means being constructed so as to load the entire set of the new parameters as an indivisible operation,

thereby constraining the reset vector to containing either the old set of parameters, or the new set of parameters, exclusively.

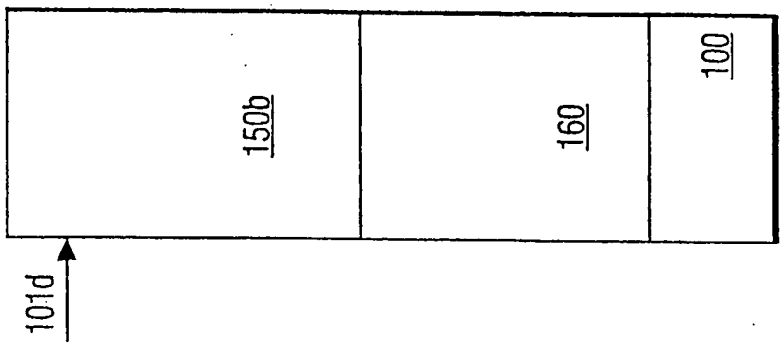


FIG. 1(d)

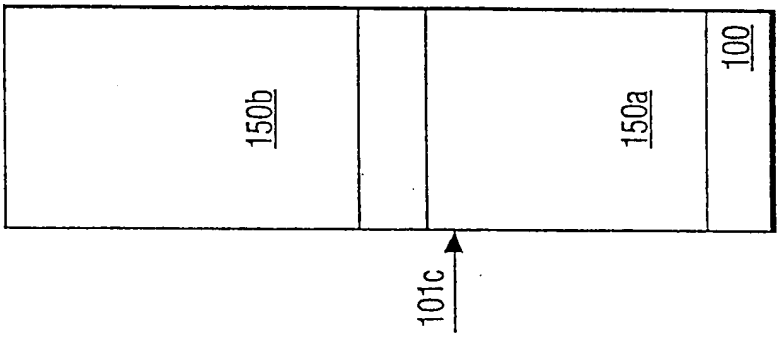


FIG. 1(c)

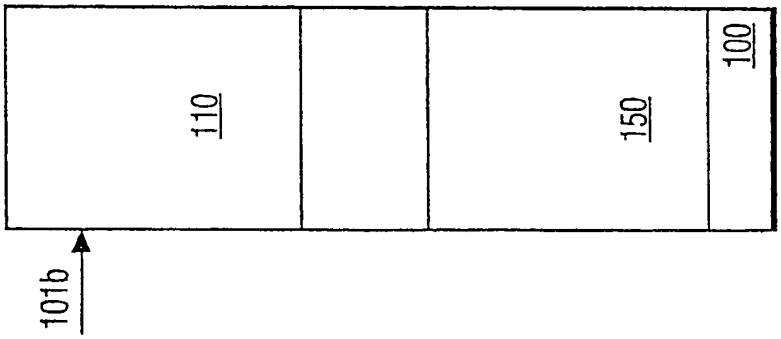


FIG. 1(b)

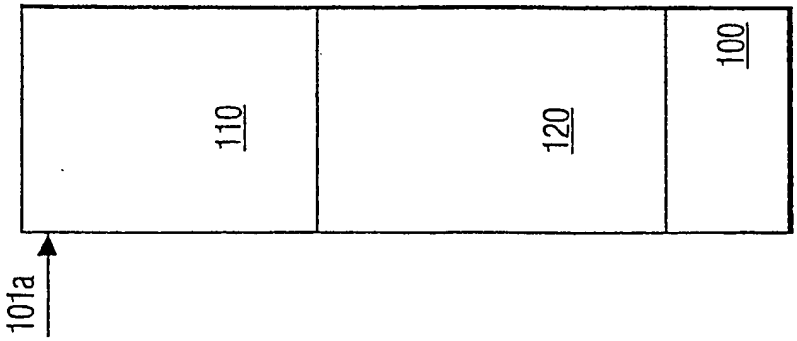


FIG. 1(a)

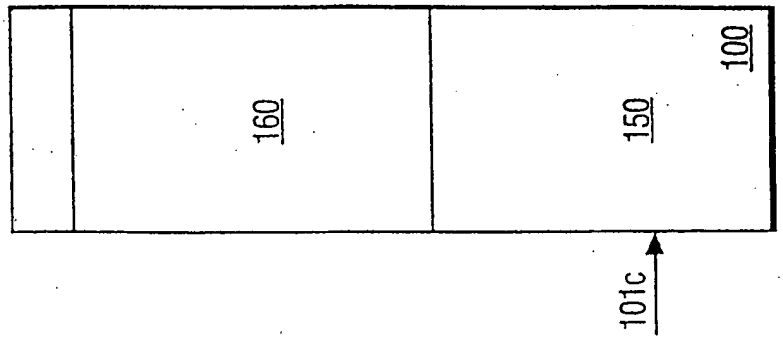


FIG. 2(c)

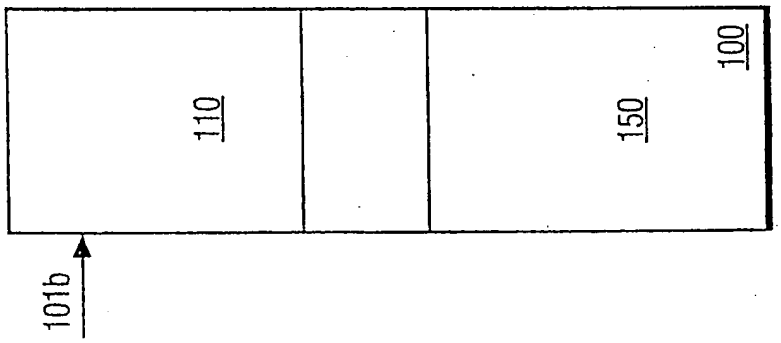


FIG. 2(b)

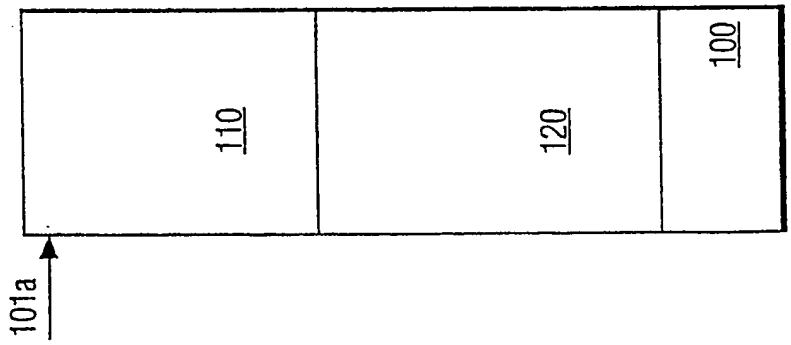


FIG. 2(a)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB 98/00334

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G06F 9/445
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0569178 A2 (AMERICAN TELEPHONE AND TELEGRAPH COMPANY), 10 November 1993 (10.11.93), column 2, line 19 - line 40	1-3,10
A	--	4-9,11,12
A	EP 0524719 A2 (DELL USA L.P.), 27 January 1993 (27.01.93), claims 1-10	1-12
	--	

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" erlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 Sept 1998	11 -09- 1998
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Göran Magnusson Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

27/07/98

International application No.
PCT/IB 98/00334

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0569178 A2	10/11/93	CA 2093042 A	09/11/93
		US 5778234 A	07/07/98
EP 0524719 A2	27/01/93	JP 6175829 A	24/06/94
		US 5388267 A	07/02/95